Cross shadowroot ARIA

Attendees:

- Joey Arhar (Google)
- Brian Kardell (Igalia)
- Valerie Young (Igalia)
- Jesse Jurman (Orthogonal Networks)
- Alex Rudenko (Google Chrome)
- Mason Freed (Google Chrome)’
- Nikki Massaro Kauffman (Red Hat)
- James Nurthen (Adobe)
- Westbrook Johnson (Adobe)
- Ben Howell (Microsoft)
- Jean-Yves Moyen (SiteImprove)
- Rob Eisenberg (Blue Spire)
- Olli Pettay (Mozilla)
- Ryosuke Niwa (Apple)
- James Craig (Apple)
- Anne van Kesteren (Apple)
- Léonie Watson (TetraLogical)
- Aaron Leventhal (Google Chrome)
-

Links

- Spring 2023 Cross-Root ARIA F2F · Issue #1005 · WICG/webcomponents (github.com)
- aom/exportid-explainer.md at exportid-explainer · behowell/aom (github.com)
- aom/semantic-delegate.md at gh-pages · alice/aom (github.com)
- [TPAC Prep] Cross Root Aria · w3c/webcomponents-cg · Discussion #68 (github.com)

Minutes

Ryosuke: if you want a relationship between an element in and outside of a shadowroot, you can do that with element references, but theres no declarative way to do that. If the element outside the shadowroot needs to reference stuff in the shadowroot, theres no way to do that right now. This session is to discuss a solution to that problem. Does anyone have suggestions on where to start?

Anne: I think ben should present on this if he is willing

Ben: I wrote a proposal about exporting IDs from the shadowroot to solve this problem.

[Explainer - [aom/exportid-explainer.md at exportid-explainer · behowell/aom (github.com)]()]

Ben: The idea in my proposal is to add a new attribute called "exportid" which would be added to an element inside the shadowroot that would label the id as being exported from the shadowroot, with some syntax to refer to it from outside the shadowroot. The goal is to allow direct referencing of elements in the shadowroot without breaking encapsulation without giving full access to the element. This should work for both closed and open shadowroots. The basis of this idea expanded on the idea of shadow parts for css styling, and this is explicitly not for css because shadow parts solves that, this is just for id references. There are other proposals that helped with affording other attributes like aria-label, so this proposal does not solve that problem. Idea is similar to how exportparts works, we dont want a big chain of id references, we want each web component to be able to control its api surface, so prevent ids from any child. For aria-labeledby you have to refer to something out of the shadowroot, this is more different from how part works. The component author would define a name that a user of the component could map with a useids. Those are the big ideas here, you could use with multiple IDs for labeledby. I also included a combobox example thats a kitchen sink of all the things put together.

Anne: Where did you introduce forward?

Ben: forwardID is not commonly used, similar to exportparts. The interesting part comes to javascript apis that return an element that is based on aria active descendant element, it parses the string and gives you the element it refers to, but that could break encapsulation. The proposal from alice that i have repurposed is to use the same retargeting algorithm that is used for events. Youll be getting the host element or whatever parent element of the actual true target that you have access to. There are some questions about what getelementbyid should return, should it do retargeting or return null? This is a point for discussion. Queryselector would return null because this is not about css. At this point it would be helpful to have some discussion.

Anne: thanks for presenting this, overall this seems pretty good to me. When we discussed this with emilio and alice, we were thinking that we only need to go inside the shadowroot and not go back out, and for the aria roles that go in reverse, we could add reverse things that people could use. For some of these, for exportID, the main use case is if you want to export multiple at the same time. If you are only exporting one, then it is better if you can target the custom element directly and you dont need to know and not have to export internals. If theres only one thing there, why would you need to be able to target it from the outside. For the user of the custom element, it should be mostly transparent how these relationships work, but some of these examples force syntax on the user of the custom element including knowing about the internals of the shadow tree which oesnt seem warranted. As for the js api, we were also considering is that we use this kind of partial selector that we would return some kind of token thing, you would get a reference to the custom element but there would be an opaque ref.

Ben: about referring only intot he shadow tree and not out, that was coupled with a proposal to add reverse, instead of aria-labeldby we could have aria-labeledfor. The problem is that it doesnt allow complicated relations like sibling shadow trees referring to each other

Anne: it would be different from how this is done, the custom element would be in charge for making the association

Ben: i havent throught through all of these and wrapped my head around it, in this instance only referring in wouldnt work in this case because there is no parent child relationship. Single direction only works for parent child

Anne: in this example, its label would refer to the child, and it would delegate its label to the inner thing, and its input would delegate to the input element thats inside it

Westbrook: one thing that this api doesnt specifically outline is that there are no one api that can cover 100% of the ax use cases that are happening. The specifics that this api is targeting is the highest complexity use case, where you have to overcome what alice has coined as the bottleneck effect. When you have x-label and x-input and there is a one to one relationship, you can bottleneck this because no two labels or associations have to happen between a shadow boundary. In the kitchen sink example, there are multiple references with multiple different elements where one thing cant map. If there is just one parent child relationship, this api is more suited to the complex solution that relates to how developers are doing these patterns today based on aria apg to make those relationships through shadow roots. I agree that an everyday developer might use a delegate for those single element case, in the case that were making complex uis and ids go in both directions are highly important to the relationships here

Brian: ben already knows this feedback, but alice asked me to pass on that its still early and could use this bikeshedding, but we really like this proposal because its really powerful and answers a lot of questions. Early proposals werent powerful and generic enough. It quickly gets unwieldy and its not great in some general common cases,
https://github.com/alice/aom/blob/gh-pages/semantic-delegate.md
^ updated alices proposal
This is a bunch of use cases for custom elements, effectively custom builtin elements, but the reason that you want the element is that you dont want that, you want your own custom shadowdom, but there is one thing inside there that is the actual element that should have the actual meaning, semantic delegate is a proposal that is an author friendly way to do that. In this is the actual input, everything should forward to that, but there are some open questions. In this combobox example, both alice and i got really lost and had to color code it because it gets really complicated. Doing complicated things gets complicated, but thats ok. The semantic delegate gets much simpler. Alice wanted me to pass along an endorsement of bens, “yes and” we would like a simple high level thing that is more author friendly

Ben: the exportID does not include semantic delegate, and imo makes semantic delegate better. exportID solves everything but can get complicated, but for cases where its simple then semantic delegate makes it simpler. Both are tools in the toolbox

Ryosuke: i think this general shape of the api looks promising, but i would prefer the original proposal with different namespace for element handles. I think that using id in this way is very confusing, it creates two different namespaces but they are using the same attribute to define the two namespaces, this is confusing. I would prefer a different name for it

Ben: we also used elementhandle, instead of exportID you would say handle, but its still using in IDref attributes, the whole point is that its an IDref. Its still mixes with ID in that scenario, and i worked with alice on this proposal and its seemed like handle wasnt adding value but was adding extra verbosity and confusing, whats the difference between a handle and an id

Ryosuke: i think that ID is still more confusing

Ben: we still have the proposal for element handle if people want to see in irc

[Proposal - RFC: Element Handles for Cross-root ARIA by behowell · Pull Request #200 · WICG/aom (github.com)]

Brian: could you say why this is better? Where is it better?

Ben: the main thing is that this is mixing in with the IDref attributes, and this syntax got confusing, and it got even more confusing in the other proposal that it was a handle, this proposal doesnt have the extra bit. It could get more verbose, you have to repeat the same thing multiple times, this is a bit less verbose. I got feedback from multiple people that the handle syntax is too verbose

Brian: i agree

Anne: this requires label to be used in a way thats different from html, you should be able to just wrap something in a label. Custom element should be able to behave like normal elements but this goes away from that by requiring people to use special syntax which is counter to the idea. If you use custom elements, you have to use this verbose AX syntax which seems like a step backwards. It should be easy to use for people like builtin html elements. That required not just looking at the relationships and the semantics. If combobox is being labels and trying to export, thats what we have to indicate somehow, then you can do label to combobox directly. If you have two labels that both target different things inside the combobox, but you do need to export something and you might need this more complicated targeting, but you should only need it for that case. Our v1 or v0 should not require making custom elements more complex. I dont think thats a good idea

Westbrook: in the last f2f in May, Apple reps requested to see examples of both the simple and the complicated (“everything”) cases

Anne: i want to solve everything, but i dont want to make the simple cases bad. This simple case should not require that syntax

Brian: if you use the verbose thing to solve the simple problem, then its going to be more verbose than the simple solution

Anne: semantic delegate wraps a lot of things together, whereas im trying to entangle those things a bit. What is the semantic role there? Youre trying to delegate the label thing, right? The combobox needs to indicate whats being labeled.

Brian: thats exactly what semantic delegate is doing

Anne: semantic delegate does other things, right?

Brian: it says “this is the input”, anything you want to do with the comboxo refers to the input

Ben: youre mentioning that we need to solve the complex problems, but that the simple problems should be simple. So we do need a way of doing this complicated syntax, but we also need a way of making things simple. With semantic delegate, they arent able to solve all of the problems, like alice’s bottleneck. Any attribute delegation proposal is complete.

Anne: i agree we need exportID, like ryosuke its nice that ID is poked through the shadow tree. Whats not clear to me is that we need the useID stuff, and whats not clear to me, well, i guess the thing is that we also need to handle the simple case well, maybe semantic delegate does that maybe not, idk. If x-combobox had a separate description inside of it and an input control and you wanted to link to it for describedby and label, that should also work without exportIDs, because exportIDs should be for realy esoteric relations where you have multiple incoming things with the same type. If the relations are from different types, you could do it through delegate defaults, this type goes to this element etc.

Westbrook: doesnt that make it unclear to the user?

Anne: thats the whole point of custom elements, the user doesnt know. We should have to explain to the consumer what the internals of the custom element looks like, thats the entire point

Brian: we should also give custom element authors the ability to export

Anne: if it's needed, if there are multiple comboboxes then yes. If the custom element represents two different addressable parts, which could be styleable parts through exportparts, but then it becomes a different thing, then you would need explicit addressing. We shouldnt require explicit addressing by default. I wouldn't feel comfortable shipping something that only has explicit addressing.

Ben: to summarize, the exportID is necessary but not complete without something simpler like semantic delegate. The proposal needs both

Anne: yes

Keith: how can i refactor my simple case to the more complex case? What would that case look like? How do i take the semantic delegate example, and add all the additional complexity? What does that look like?

Brian: if you click on the link in the thing there are side by side examples https://github.com/alice/aom/blob/gh-pages/semantic-delegate.md. The bottom one becomes the top one more or less.

Anne: its not easy when you have to scroll to see the whole thing, the examples are complex

Westbrook: the kitchen sink is the one that has to have this complexity. As much research has been done for semantic delegate, only the exportID apis have proved to be actually capable of creating the AX tree thats required

Anne: im not trying to dismiss the example, just that the example is easy

Michael Warren: quick question and comment. It seems like its showing that the export case can only go one level up, each element delegates exactly one level up. Is that the case? That the delegation only goes up one level? Would it be simpler if the export was multi level, to a host with an id, so you wouldnt have to directly reference. If you could export several host levels instead of one, would that simplify the api?

Ryosuke: web components should not want to do that from the encapsulation perspective

Anne: for that we want forwarded IDs, which is the same solution that shadowparts had, and that allows preserving the encapsulation for each shadow tree.

Ryosuke: design principle of shadowdom is that the custom element controls what is exported. Author has control over what is exported

Ryosuke: this is one solution we have discussed. Are there other proposals worth discussing right now?

Anne: i think this might be the only one that is actively being developed. This is a continuation of what we talked about at the igalia F2F.

Westbrook: if theres no other questions, we could look at how the semantic delegate api does things that arent attribute relationships. An interesting thing to support as a web developer if this was taken on. Would that be useful? One api that requires scrolling, a similar piece of AX that needs to be made is parent child relationship AX. in this example, we are making a radiogroup, and we are attempting to use custom elements. We would like to use buttons inside our custom elements, but we want to encapsulate style, so we created an x-button. Because of this, the parent child relationship is no longer achievable with all of the requirements that ive outlined. In theory, today we could make x-button do all of the stuff that the button element does. We could keep this relationship by making the x-button a semantic delegate. In this example we are not using any ID reference, but we are allowing the semantic delegate to do stuff across the shadowroot.

Jesse Jurman?: That isn’t referring to an ID or a button tag?

Westbrook: by ID, you refer to an element child. It refers to an element inside the shadowroot. This is an rea where the api that ben is proposing, while powerful, doesnt handle the full complexity of the shadowroot. Semantic delegate covers things that are simple, and this is something that we can solve in a simple way . the only person who needs to know is the component developer, and the consumer doesnt need to know. This could also play into relationships in lists and tables. A lot of capabilities are opened up with this api.

Brian: the input is used there to make the example clear, but it because it serializes nicely. Theres an example thats linked

Westbrook: here is the imperative version of this api, thanks for calling it out brain

Anne: it feels weird that the shadworoot has a delegate and not the custom element

Westbrook: without a shadowroot, you dont need a delegate. If it was in the lightdom you would know the content because you would put it there

Anne: im saying that it seems weird for the relationship to live on the shadowroot. Elementinternals could reference anywhere

Brian: ah so you’re saying “ instead of being on shadowroot it should be in element internals?”

Anne: yes.

Keith: what if youre assigning a delegate to a custom shadow root?

Anne: i guess it depends on what it refers to? Maybe thats a bug? I dont know. You need to define it anyway, this input element would/could be another element. Theres no guarantees that its actually in the shadowroot

Keith: is it a good justification to have an imperative api for this?

Brian: who would write it? You could do it if you use a template. It belongs to the author definition, not the use, so i think thats why - the custom element author

Keith: the template belongs to the custom element author as well

Anne: i was wondering, a lot of the examples involve buttons, what if elementinternals support buttons? That feels like an easy extension we could make. Like on elementinternals, you would say this representas a button or a submit button. Like building on form associated custom elements. And then the user agent would treat it as a button, for like a variety of purposes. It would be a custom element but have button element semantics

Ryosuke: we dont have enough time for the rest of this discussion. Whats the next step here?

Ben: i am working on a prototype in chromium of this api. I think it sounds like the general feedback is positive about exportid but that its incomplete without something for thesimple case. I think it might be worth continuing the prototype while trying to merge the proposals for one use together.

Anne: that summary doesnt capture the feedback. In particular, its important to distinguish semantic delegate groups the types and puts them in a single thing, custom elements either have incoming relationships like delegates, and those are different types, and only when there are multiple incoming of the same type is when youd need exportID, but for the others y0ou should be able to export by type.

Brian: can we say that there are two broad solutions/problems?

James: one of the problems is that the most opinionated stakeholders are in different timezones. Theres no time where the 5 or 6 of you are in the same room. Ben you could arrange that with other stakeholders? Anne, Alice, Westbrook, etc…

Ben: yes, trying to figure out the best forum for that

James: we could use the same AOM list but not PT Afternoon

Ryosuke: with that we can end this session. Thanks everyone!